The Fat Pencil, the Cocktail Napkin, and the Slide Library

Mark D Gross
College of Architecture and Planning
University of Colorado
Boulder, CO 80309-0314
mdg@cs.colorado.edu

Abstract

The paper describes recent explorations in sketch recognition and management to support architectural design. The exploration and decision-making of early, conceptual design better suited to freehand drawing, sketching, and diagramming than to the hard-line drawing and construction kit approaches of traditional CAD. However, current sketch programs that simulate paper and pencil fail to take advantage of symbolic manipulation and interactive editing offered by computational environments. The paper presents a ‘computer as cocktail napkin’ program, which recognizes and interprets hand-drawn diagrams and provides a graphical search facility, simulated tracing paper, and a multi-user shared drawing surface. The cocktail napkin is the basis of Stretch-A-Sketch, a constraint based draw program that maintains spatial relations initially specified by a diagram. The cocktail napkin program is also the basis for a query-by-diagram scheme to access a case based design aid as well as to a small collection of images of famous buildings. The paper briefly reviews these extensions of the cocktail napkin program.

Introduction - The Fat Pencil

Anthony Pellechia, an architect who worked in Louis Kahn's office, tells a story of Kahn and his fat pencil:

> *Lou had a tendency to always work with the charcoal; and you had to watch ou for him, because he cheated a lot. That charcoal line was very thick, and if you're working at an eighth or a sixteenth, you always had to keep putting a scale on it, because he was really very capable of camouflaging the scale of it. He would make everything work and then he'd go away. You wouldn't see him for maybe the next day, and you were left with these very thick lines that when reduced to realistic wall thicknesses and spaces — you couldn't put this functional stuff back in.* [1], p 52.

In the early, conceptual, phase of designing, many designers work with soft pencils or thick markers on cheap yellow tracing paper. These tools - rough and ready - enable the designer to explore alternatives and variations with great speed and low commitment. The imprecision of the tools keeps the designer in mind of the conceptual nature of the exploration, and it helps the designer avoid the temptation of making decisions at a finer level of detail than appropriate. The interplay of loose conceptual sketches and more definite schematic drawings is discussed in recent examinations of the role of architectural drawing in design [2, 3].

Regrettably, computer aided architectural design tools do not support this early conceptual designing. Most CAD programs require commitment to precise shape, position, dimension, lineweight and color, and impose a highly structured procedure for making and editing drawings. For example, drawing a circle in AutoCAD is a four step process. One must (1) choose the "drawing tools" menu, (2) select one of several ways to draw circle (e.g. center point and radius) from the menu, (3) identify the center point, (4) identify a point on the radius. By this time, the designer with an HB pencil has drawn circles around AutoCAD. AutoCAD's overhead may be worthwhile in later phases of design, when one is certain that it is a circle one wants, and reasonably sure of its dimension and position. The advantage of the structured drawing process is that it results in a structured representation that permits, for example, the application of energy, lighting, and structural simulations, as well as editing. But in conceptual design, the structured drawing process of most CAD programs inhibits the flow of thought and demands more precision than the designer is ready to commit to.

In contrast to AutoCAD and similar structured CAD editors, freehand drawing programs (such as Fractal Painter), provide users with a simulation of sketching on paper. Designers who master these programs find the ability to draw freehand combined with the editing capabilities and "effects" of digital media advantageous. However, these programs suffer from the reciprocal deficiency: the lack of a structured representation for the design. It is difficult to move smoothly from conceptual design sketches to schematic design CAD drawings, because pixel-based sketching programs do not support any representation of the design's elements. Therefore the designer must re-enter the design that was originally sketched freehand in the more rigidly constrained environment of a three-dimensional modeler or CAD program.

Conceptual design demands the best of both worlds. On the one hand, the rapid and unstructured exploration of line, shape, and configuration; on the other hand, the representation of elements, attributes, and relations that makes possible editing and other computation on designs. In short, design tools should support the freehand sketching and diagram-making that characterize conceptual designing in a way that enables the designer to move smoothly to the more precise representations used during schematic design. The "Electronic Cocktail Napkin" program is an experimental prototype built to explore this idea and advance this goal.

The remainder of this paper is organized as follows. Section 2 describes the Electronic Cocktail Napkin, a pen-based drawing program that reads, recognizes, and manages hand drawn diagrams. The recognition algorithm is outlined and the cocktail napkin's support for graphical search, simulated tracing paper, and multi-user shared drawing is described. Sections 3 and 4 describe extensions to the basic cocktail napkin. Section 3 describes Stretch-A-Sketch, a constraint based drawing program that maintains spatial relations identified in a diagram. Section 4 describes two efforts to index databases of designs with diagrams. Section 5 concludes with a summary, a discussion of related work and directions for further research.

2. Electronic Cocktail Napkin

2.1 Recognizing hand-drawn glyphs

The Cocktail Napkin program [4] is an interactive, trainable recognizer for hand-drawn diagrams, written in Macintosh Common Lisp 2.0 . As with a typical pen-computing environment, the user simply draws on a tablet, making marks in the drawing window. The program tries to recognize the designer's marks as letters, shapes, and lines. The program also has higher-level recognition facilities that can identify configurations of the simple shapes; for example, a "tree diagram" or a "floorplan bubble diagram." In addition, the program affords a simple simulation of tracing paper, support for two simultaneous users, and search for configurations. The cocktail napkin interface (figure 1 below) includes a sketching window, a message and type-in area where the program interacts textually with the user, command buttons, trace tabs to select sheets of simulated tracing paper, and a catalog of previously made diagrams and layers of trace.

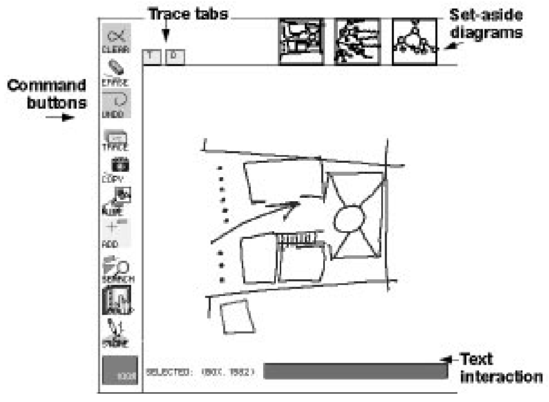
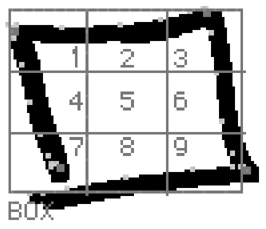


Figure 1. The cocktail napkin interface.

The cocktail napkin program reads x, y, and pressure values from a Wacom SD series tablet at 1200 baud. The input data is parsed as a series of simple symbols, or glyphs. A glyph begins when the user touches the pen to the tablet and ends when the user lifts the pen for more than a certain duration (by default 1/4 second). A glyph is characterized by the pen path through a 3x3 grid inscribed in the glyph's bounding box, the number of strokes, the number of corners, and the size and aspect ratio of the bounding box (figure 2).



```
GLYPH -- BOX.1676
TYPE BOX
NSTROKES 1
NCORNERS 4
PEN-PATH (7 4 1 2 3 6 9 7)
SIZE LARGE
ASPECT RATIO SQUARE
LOCATION NIL
ROTATION NIL
```

Figure 2. Identifying characteristics of a simple glyph.

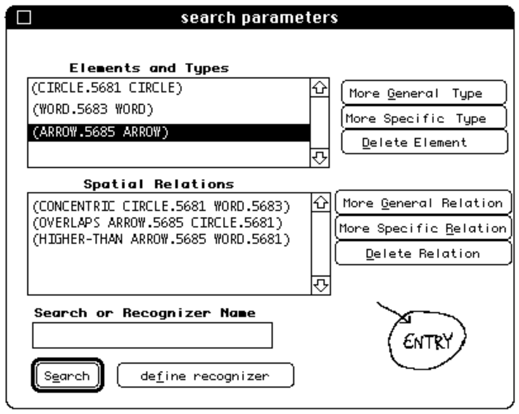
Once these characteristics of a glyph have been identified, the input glyph is compared with a library of previously trained glyph templates. If a single template matches the input glyph, then the input glyph is classified; if more than one template matches, then either the ambiguity is carried or the user is asked to resolve it. The present version is trained with capital letters, numbers, and simple shapes (circles, boxes, triangles, and various types of lines). This recognition algorithm suits the purpose: it is interactively trainable, multi-stroke, simple to implement, and accurate enough for an experimental prototype. Other, more intricate, algorithms have been developed [5-7] and could replace the simple version employed here.

It is easy to train the cocktail napkin to recognize a new glyph. The user first draws a collection of examples that serves as a training set for the new glyph template. The characteristics described above are extracted from each and stored in the new template. In most cases ten to fifteen samples are sufficient to uniquely identify a new glyph. The recognition algorithm can work interactively, or it can work in batch mode, waiting until the user stops drawing before trying to identify glyphs. Once the program identifies an input glyph, it can display it either in rectified form (a crude round shape becomes a circle or oval; a rough line becomes straight) or as drawn by the user. By default the diagram remains in "as drawn" form.

2.2 Recognizing glyph configurations

In addition to identifying single symbols, or glyphs, the program also recognizes arrangements of glyphs in certain spatial relations. For example, to work with bubble diagrams of floorplans, the cocktail napkin can be programmed to identify a circle or box containing a word as an instance of a room.

The cocktail napkin identifies binary spatial relations among pairs of selected glyphs, for example 'contains,' 'above,' 'near,' 'connects.'   The program's analysis of spatial relations appears in a "search" dialog that lists glyph types in the configuration and the spatial relations among them. For example, figure 3 shows the program's analysis of a simple configuration of an arrow, a circle, and a word.



```
(CIRCLE.5681 CIRCLE)
(WORD.5683 WORD)
(ARROW.5685 ARROW)
```

```
(CONCENTRIC CIRCLE.5681 WORD.5683)
(OVERLAPS ARROW.5685 CIRCLE.5681)
(HIGHER-THAN ARROW.5685 WORD.5681)
```

Figure 3. Search dialog.

Descriptions in the search dialog are used to control graphical search for similar configurations, and to define higher-level recognizers. The buttons in the search dialog adjust the program's description of the configuration. The 'delete' buttons enable the user to eliminate an undesired element or a spatial relation from the description. For example in figure 3, we might delete the relation that states the arrow is higher-than the word. The 'general' and 'specific' buttons enable the user to make the description of a glyph type or a spatial relation more general or more specific. For example, a glyph identified as a 'circle' may be described more generally as a 'shape'; the 'contains' relation may be described as the more specific 'concentric.'

2.3 Tracing Paper

Recognizing glyphs and configurations of glyphs and spatial relations is the backbone of the cocktail napkin program. However, the program has provided a starting place for several experimental extensions, which are described in the remainder of this section and in sections 3 and 4, following.

Tracing paper, (and its CAD surrogate, "layers") plays an important role in design. Esepcially in conceptual design, designers use layers of tracing paper to copy, excerpt, reposition and rotate, and combine fragments of design alternatives. (In structured CAD programs designers can assign elements to different layers, for example, placing roof elements on one layer and wall elements on another; but this is a different use of layers.) At the risk of mixing metaphors, the cocktail napkin program includes a simple simulation of tracing paper. Our implementation of simulated tracing paper adheres to the constraints of real trace -- one can move any layer of trace, but draw and erase only on the top sheet; figures on lower sheets fade away as new trace is placed on top. Layers of trace can be removed from the stack and set aside.

When the designer places a new sheet of trace on the drawing (either by selecting the trace command from the menu of icons or by making the 'trace' gesture (three horizontal lines stacked vertically) the program shades the drawing area, fades the existing glyphs, and adds a 'trace tab' to the list on the top left of the screen (see figure 1); '0' is the lowest layer and 'T' the top, active drawing, sheet . The trace tabs enable the designer to select a trace sheet. The designer can trace glyphs from lower layers by hand or use the 'copy' command to transfer them selectively to the top sheet. A trace sheet may be selected and shifted beneath the top layer, or removed and set aside. For want of screen space, sheets of trace set aside appear in reduced size in a catalog along the top of the screen (see figure 1).

2.4 Sharing the Drawing Surface

Designing is seldom done alone, and often designers share a drawing surface [8]. The cocktail napkin metaphor suggests several designers around the table, each with pen in hand, taking turns at drawing on a napkin. To support this "shared drawing space" two different prototypes were built. In the first prototype, designers share a single drawing pad; in the second, designers work at separate tablets but share the same drawing space. Currently a great deal of work is being done on multi-user drawing environments [9-12]; the innovation here is to combine a multi-user drawing with the recognition features of the cocktail napkin.

In the first prototype where designers share a single tablet (figure 4a), we adjusted the resonant frequency of two Wacom pens to give each designer's pen a unique electronic signature. The tablet hardware limits drawing to one pen at a time. When either pen is detected near the tablet, the program determines which designer is drawing, adds an 'author' tag to the glyphs, and displays glyphs drawn by each designer in a different color.

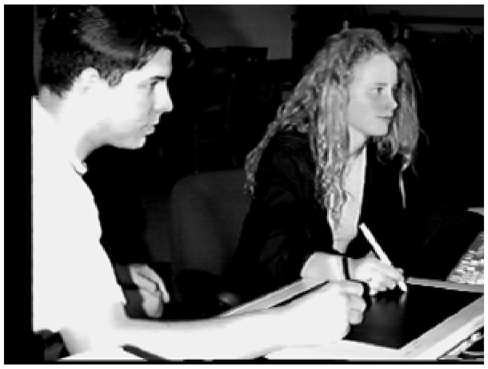
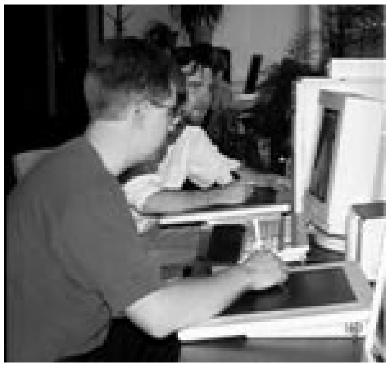



Figure 4a. Two designers sharing a single tablet.

Figure 4b. Two designers sharing a drawing space but using their own tablets.

In the second prototype, we connected two Wacom tablets to the 'a' and 'b' serial ports of a single computer, and added a few lines of code to the low-level driver program to serve both serial ports.   The two designers share the virtual drawing space, but work separately (figure 4b). Again, the program tags each glyph with its author and displays different colors for each designer.

When two or more designers work together on a drawing, often a spoken conversation provides essential communication about what is going on. Merely examining the drawing, after it is done, may have little meaning for anyone other than the orginal participants. But if others could observe the sequence in which the drawing was made <u>and</u> hear the conversation, perhaps the drawing could be understood. To explore this idea we programmed the built in Macintosh sound recording facilities to eavesdrop on the designers’ conversation. We tag the sound bytes with identifiers that connect elements of the diagram with the conversation that was going on at the time. A step-by-step replay function redisplays the evolving drawing along with the relevant conversation fragment.

3. Stretch-A-Sketch - Drawings That Behave

Stretch a Sketch [13], is an experimental prototype that adds interactive constraint based behavior to the diagrams managed by the cocktail napkin program. Previous work [14-16] has explored the construction of constraint based drawing programs, which enforce and maintain spatial relationships (adjacencies, proportions, minimum and maximum dimensions, connections) among drawing elements. The computer maintains desired relations as the designer changes the positions and sizes of drawing elements.   In addition to spatial relations, constraints may describe functional desiderata, such as structural strength and stability, lighting, or energy performance.

Constraint based drawing programs have suffered from user-interface overhead. The user must explicitly add constraints to describe desired spatial relations. For example, in CoDraw, the designer must select elements to be related, then choose a spatial relation from a ‘constraints’ menu. This step further distances the designer from the design.

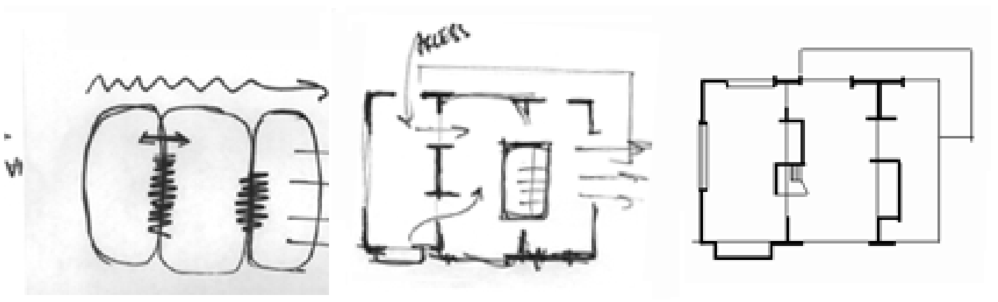


Figure 5. Design drawing sequence (with traditional media) increases in specificity.

Stretch-A-Sketch takes advantage of the fact that designers proceed from rough diagrams made in conceptual design to more precise drawings in schematic design, carrying along the same basic spatial relations (figure 5). Stretch-A-Sketch identifies the spatial relations in a hand drawn diagram, and maintains these relations as the user moves and resizes diagram elements and moves gradually to more precise CAD-like drawing and editing.

Stretch-A-Sketch breaks the metaphor of the cocktail napkin, by permitting the designer to edit (move and resize) diagram elements. By holding down a button on the pen-barrel, the designer puts the pen into 'command mode.' Then diagram elements can be selected, moved, and resized.

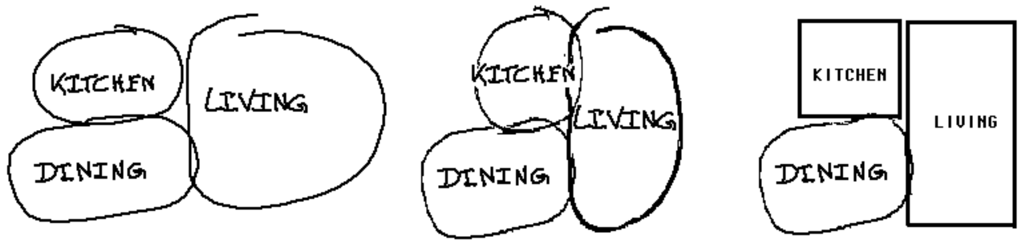


Figure 6. Stretch-A-Sketch maintains relations in the diagram. (a) diagram as first drawn; (b) after resizing a room; (c) partially rectified drawing.

When the designer selects an element to be moved or resized, Stretch-A-Sketch first determines the spatial relations it engages in. For example, a ‘box’ may be to the right of one element, contain another, and below a third. After the designer has moved or resized the box, Stretch-A-Sketch reasserts the spatial relations, adjusting the sizes and positions of related elements to satisfy the relations again.

4. Diagrams as Indices to Design Databases

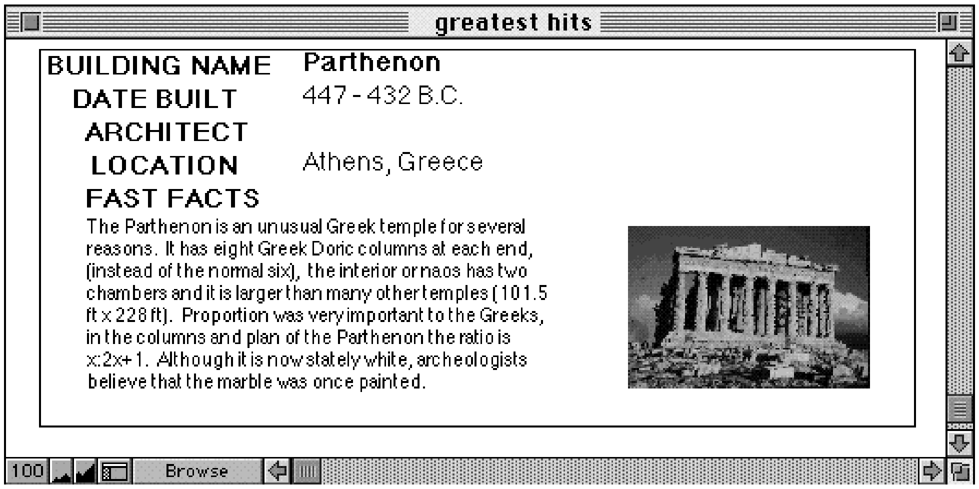
4.1 Why index with diagrams?

Most sources of information useful to designers are indexed textually, by key words. For example, visual collections (slide libraries) are indexed by key words that describe architect, date, location, and name of building. The Getty Art and Architecture Thesaurus [17] provides an additional range of key words to describe image content, classifying a building, for example, as a “prairie style wood frame residence”. Likewise, information sources that document specific buildings or building types, for example post-occupancy evaluations (POE) are organized by key words. Given the dominant role that visual thinking plays in architectural design, we decided to index these information sources visually, using diagrams to formulate queries into a database. If diagrams could trigger retrieval of relevant information, we could build critiquing mechanisms that watch a developing design and provide critiques and precedents relevant to the work at hand.

We have built two prototypes to retrieve database items using diagrams as index keys. In the first prototype [18], diagrams are used to index a collection of images of famous buildings. In the second prototype [19] diagrams index a case base of post-occupancy evaluations of courthouse and library designs.

4.2 The Slide Library

We constructed a small visual collection of well-known buildings, built in FileMaker Pro, a database program. Each record includes a photograph of the building, textual information describing the building, architect, date and a brief description of the building (Figure 7).



BUILDING NAME	Parthenon
DATE BUILT	447 - 432 B.C.
ARCHITECT	
LOCATION	Athens, Greece
FAST FACTS	

The Parthenon is an unusual Greek temple for several reasons. It has eight Greek Doric columns at each end, (instead of the normal six), the interior or naos has two chambers and it is larger than many other temples (101.5 ft x 228 ft). Proportion was very important to the Greeks, in the columns and plan of the Parthenon the ratio is x:2x+1. Although it is now stately white, archeologists believe that the marble was once painted.

Figure 7. Filemaker Pro database of architecture's "greatest hits."

We indexed each building with one or more simple diagrams, obtained from architects and architecture students. In two pilot experiments [20], we found that (a) architecture students make similar diagrams from drawings and photographs, and (b) designers made similar diagrams from memory of well-known buildings. By 'similar' we mean the diagrams contained many of the same graphical elements organized in the same ways. For example, figure 8 shows drawings made from memory of Wright's Guggenheim museum. Most of the diagrams contain either a set of stacked rectangles or a spiral.

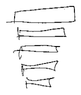
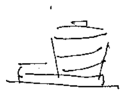
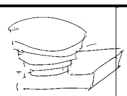
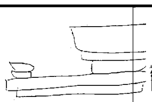
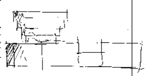
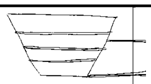
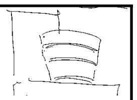
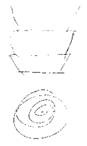
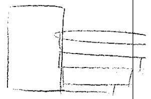
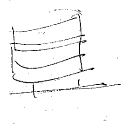
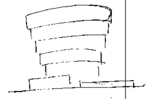
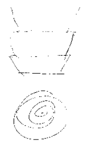
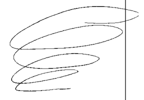
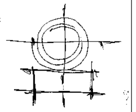
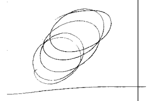
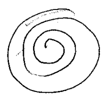
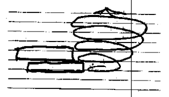
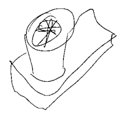
STACKED BOXES							
STACKED BOXES (cont'd)							
SPIRAL							
MISC.							

Figure 8. Designers' diagrams of Wright's Guggenheim museum, from memory.

The designer issues a visual query by drawing a diagram. The cocktail napkin program first parses the query into a symbolic description of glyphs and spatial relations (figure 9). A simple matcher then compares the query diagram with the stored diagram keys that index the database. The program identifies the database records whose keys most closely resemble the query, and sends a message (an AppleEvent) to the FileMaker Pro process to display those records.

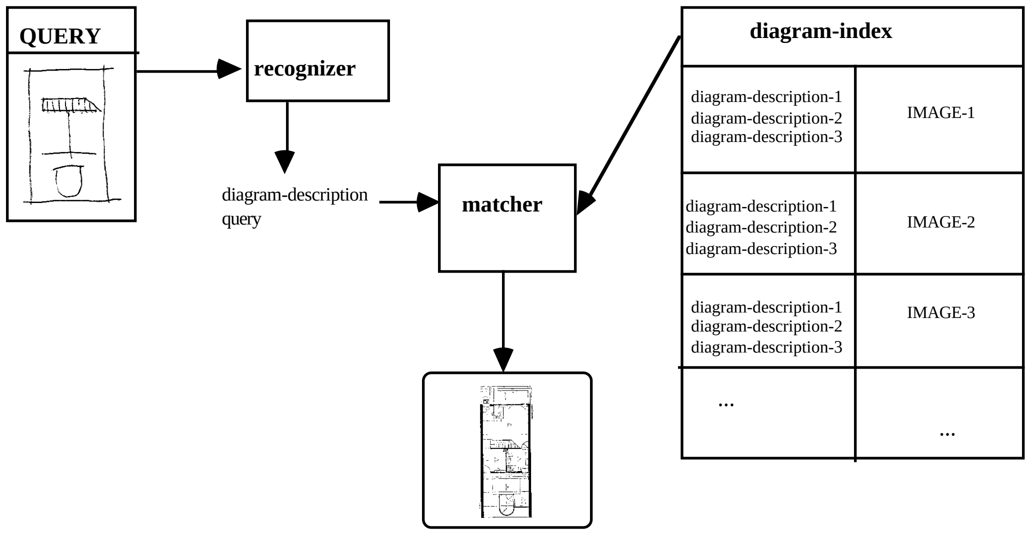


Figure 9. Functional outline of the query by diagram system.

4. 3 Diagram index to Archie - A Case Based Design Aid

Archie is a case based design aid for architecture developed by colleagues at Georgia Tech [21]. It contains post-occupancy evaluation (POE) data on libraries and courthouses. POE data in Archie is organized in the form of problems, responses, and stories from real world cases. The data is indexed by key words in a number of categories: systems (e.g. circulation, mechanical), roles (client, contractor, designer), etc. A generic case based design aid shell (CBDA) provides a framework for locating this information, and tools for indexing and accessing relevant problems, stories, and responses.

We constructed a diagrammatic index to Archie, similar in structure to the diagram query scheme for the visual collection [19]. Many items in the case base include graphics -- photos or plan drawings. We decided to augment these with simpler diagrams that illustrate the key points. For example, a story about "the courtroom is connected with the waiting area by a corridor" was diagrammed with a modified 'dumbell' - two circles connected by an arrow and a circle in the middle (figure 10). Many architectural stories can be usefully annotated by similarly simple diagrams (see for example the work of Lynch [22]). Why not include the diagrams as indices? The diagram in the Archie case base serves as a visual reminder of the index key.

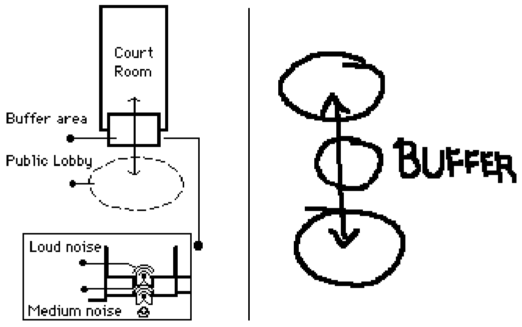


Fig. 10. An illustration from a courthouse story with corresponding diagram.

The scheme for linking Archie with the cocktail napkin program is similar to that used with the visual collection. An index table pairs diagrams with items in the case base. The designer draws a visual query, and the program returns a list of items whose diagrams match the query. The cocktail napkin program then requests Archie to display those items.

5. Discussion

5. 1 Related work

Outside of CAD, three main research communities are interested in the issues discussed in this paper: the visual languages (VL) community, the computers and human interface (CHI) community, and the artificial intelligence (AI) community. Lakin's work on parsing visual languages most vividly describes the requirements for computational visual languages in design [23, 24]. Other relevant work in the visual language community includes Futrelle's work on diagram understanding[25] and work on parsing visual languages [26, 27]. Work in computers and human interace (CHI) includes the work on recognition cited above, [6, 7], graphical search [28], shared drawing spaces [8, 9, 11, 12]. In the AI community, interests center on visual representations and their role in reasoning [29].

Research on sketching and diagramming has also been done in the domain of architecture and environmental design. More than twenty years ago, MIT’s Architecture Machine Group was beginning to explore sketch recognition technologies [30]; however these explorations were supplanted by other topics in computer graphics and human-machine interaction. More recently, Ervin's work on the structure and function of diagrams in design develops the relationship between constraint based programming and spatial relations in diagrams [31]. His CBD program constructs a diagram from a description of spatial relations (constraints);

once constructed, the diagram retains its essential characterstics. Other programs that construct diagrams from symbolic descriptions have also been constructed [32] Harrison's recent article profiling the work at Xerox Palo Alto Research Center (PARC) emphasizes both the centrality of hand drawing and gesture as well as the role of collaborative work in design [8].

5.2 Summary and Conclusions

What began as a simple recognizer for hand drawn shapes developed into a collection of diverse, but related projects in sketch recognition and management. This paper has given an overview of these explorations. However, with this diversity of topics, the main points threaten to become lost.

In summary, we have argued that direct hand drawn input in the form of sketches and diagrams is necessary to support conceptual design, but that sketching must be coupled with recognition and structured CAD representations in order to facilitate design development into the schematic phase.   If pen input can be coupled with traditional structured representations of designs, then analyses (e.g. critiquing, simulation) can be provided earlier in the design process when it can make the most difference. Recognizing key configurations in a hand drawn drawing can trigger reminding of relevant examples in a case base or a visual collection that has been diagrammatically indexed. Finally, diagrams in conceptual design often embody key elements and relationships that persist through later stages of design. Stretch-

A-Sketch pursues this idea, establishing as constraints relationships found in the hand drawn diagram, which are maintained as the designer later edits the drawing.

Although the explorations describe here are at a tentative stage, and the programs are but working prototypes, the results so far have been encouraging. Future stages of the project will include refining the recognition algorithm and enhancing the basic cocktail napkin features, expanding the size of the databases indexed, and performing real sketching experiments with the shared-drawing surface. A central limitation is the tablet hardware, which requires users to look at the display screen while drawing on the tablet. This is disconcerting for new users and requires several hours to overcome. We plan to replace the tablet hardware with a touch sensitive LCD display.

ACKNOWLDGEMENTS

Jim Davis (Cornell Design Research Institute), Aaron Fleisher (MIT), and Kyle Kuczun (of our group) all provided helpful arguments at various stages of the prototype development. Ellen Do and Craig Zimring (of Georgia Tech) collaborated on the development of the Archie index. Kristin Mayfield (an undergraduate student in our group) collaborated on the slide library index project. We gratefully acknowledge support from NSF grant #93-13186 which supported the development of constraint management code used in Stretch-A-Sketch.

References

1. MIT, *Processes in Architecture.* 1979, Cambridge, MA: MIT School of Architecture and COmmittee on the Visual Arts.

2. Fraser, I. and R. Henmi, *Envisioning Architecture - an analysis of drawing*. 1994, New York: Van Nostrand Reinhold.

3. Herbert, D., *Architectural Study Drawings.* 1993, New York: Van Nostrand Reinhold.

4. Gross, M.D. *Recognizing and Interpreting Diagrams in Design*. in *Advanced Visual Interfaces.* 1994. Bari, Italy:

5. Apte, A., V. Vo, and T.D. Kimura. *Recognizing Multistroke Geometric Shapes: An Experimental Evaluation*. in *ACM conference on User Interface and Software Technology (UIST)*. 1993. Atlanta, GA: ACM.

6. Rubine, D., *Specifying Gestures by Example.* Computer Graphics, 1991. **25**(4): p. 329-337.

7. Zhao, R. *Incremental Recognition in Gesture-Based and Syntax-Directed Diagram Editors.* in *INTERCHI '93*. 1993. Amsterdam: ACM / Addison Wesley.

8. Harrison, S., *Computing and the Social Nature of Design.* ACADIA Quarterly, 1993. **12**(1): p. 10-18.

9. Bly, S., S. Harrison, and S. Irwin, *Media Spaces: Bringing People Together in a Video, Audio, and Computing Environment.* Communications of the ACM, 1993. **36**(1): p. 26-45.

10. Ishii, H. and N. Miyake, *Toward an Open Shared Workspace: Computer and Video Fusion Approach of Team Workstation.* Communications of the ACM, 1991. **34**(12): p. 37-50.

11. Tang, J.C. and S.L. Minneman. *VideoWhiteboard: Video Shadows to Support Remote Collaboration.* in *CHI '91*. 1991. New Orleans, LA: ACM Press / Addison Wesley.

12. Wellner, P., *Interacting with Paper on the Digitaldesk.* CACM, 1993. **36**(7): p. 87-96.

13. Gross, M.D., *Stretch-A-Sketch, a Dynamic Diagrammer*. 1994, College of Architecture and Planning, University of Colorado, Advanced Computing Projects Lab:

14. Gross, M., *Relational Modeling,* in *The Electronic Design Studio,* M. McCullough, W. Mitchell, andP. Purcell, Editor. 1990, MIT Press: Cambridge, MA. p. 123-136.

15. Gross, M. *Grids in Design and CAD*. in *ACADIA '91*. 1991. Los Angeles:

16. Gross, M.D. *Graphical Constraints in CoDraw*. in *IEEE Workshop on Visual Languages*. 1992. Seattle: IEEE Press.

17. Trust, J.P.G., *Art & Architecture Thesaurus*. 1990, New York: Oxford.

18. Gross, M.D. *Indexing visual databases of designs with diagrams*. in *Design Decision Support Systems*. 1994. Vaals, Netherlands:

19. Gross, M.D., C. Zimring, and E. Do, *Using Diagrams to Access a Case Base of Architectural Designs,* in *Artificial Intelligence in Design '94,* J. Gero, Editor. 1994, Kluwer: Lausanne.

20. Gross, M.D., *Visual Query for Visual Databases*. 1994, College of Architecture and Planning, University of Colorado, Advanced Computing Projects Lab:

21. Domeshek, E. and J. Kolodner, *A case-based design aid for architecture,* in *Artificial Intelligence in Design,* J.S. Gero, Editor. 1992, Kluwer: Netherlands.

22. Lynch, K., *Image of the City*. 1965, Cambridge, MA: MIT Press.

23. Lakin, F. *Visual grammars for visual languages*. in *National Conference on Artificial Intelligence (AAAI).* 1987.

24. Lakin, F., *et al.*, *The electronic notebook: performing medium and processing medium.* Visual Computer, 1989. **5**: p. 214-226.

25. Futrelle, R.P. *Strategies for Diagram Understanding: Generalized Equivalence, Spatial / Object Pyramids and Animate Vision*. in *10th Intl Conf on Pattern Recognition.* 1990. IEEE Computer Society Press.

26. Wittenburg, K. and L. Weitzman. *Visual Grammars and Incremental Parsing*. in *IEEE Workshop on Visual Languages*. 1990. Skokie, IL:

27. Golin, E., *A Method for the Specification and Parsing of Visual Languages*. 1991, Brown University:

28. Kurlander, D. and E. Bier. *Graphical Search and Replace.* in *SIGGRAPH.* 1988. Atlanta, GA: ACM Press.

29. Chandrasekaran, B., N.H. Narayanan, and Y. Iwasaki, *Reasoning with Diagrammatic Representations.* AI Magazine, 1993. **14**(2): p. 49-56.

30. Negroponte, N. *Recent advances in sketch recognition*. in *AFIPS (American Federation of Information Processing) National Computer Conference*. 1973. Boston, MA:

31. Ervin, S.M., *Designing with Diagrams,* in *The Electronic Design Studio: Architectural Knowledge and Media in the Computer Age,* M. McCullough, W.J. Mitchell, andP. Purcell, Editor. 1990, MIT Press: Cambridge, MA. p. 107-122.

32. Dave, B., *CDT: A Computer Assisted Diagramming Tool,* in *CAAD Futures '93 - Proceedings of the Fifth International Conference on Computer-Aided Architectural Design Futures,* U. Flemming and S.v. Wyk, Editor. 1993, North-Holland: Amsterdam. p. 91-109.